

## CLAIMS

- 1     1.     A method for executing uniprocessor (UP) coded workloads in a multiprocessor  
2     (MP) computer system without having to rewrite the UP-coded workloads' code, the  
3     method comprising the steps:  
4         organizing the UP-coded workloads into one or more concurrency groups,  
5     wherein UP-coded workloads in the same concurrency group are not permitted to execute  
6     concurrently with one another in the MP computer system;  
7         scheduling first and second execution vehicles that respectively execute on differ-  
8     ent processors in the MP computer system at substantially the same time;  
9         acquiring a first concurrency group by the first execution vehicle and a second  
10     concurrency group by the second execution vehicle; and  
11         executing UP-coded workloads in the first concurrency group through the first  
12     execution vehicle at substantially the same time as UP-coded workloads in the second  
13     concurrency group are executed through the second execution vehicle.
- 1     2.     The method according to claim 1, wherein the UP-coded workloads are UP-coded  
2     threads, and the first and second execution vehicles are first and second processes.
- 1     3.     The method according to claim 1, wherein the UP-coded workloads are messages,  
2     and the first and second execution vehicles are first and second threads.
- 1     4.     The method according to claim 1, wherein the step of acquiring the first and sec-  
2     ond concurrency groups further comprises:  
3         dequeueing from a concurrency-group run queue a first concurrency-group data  
4     structure associated with the first concurrency group; and  
5         dequeueing from the concurrency-group run queue a second concurrency-group  
6     data structure associated with the second concurrency group.
- 1     5.     The method according to claim 4, further comprising:

2           setting a first CG flag in the first concurrency-group data structure to a value indi-  
3           cating that the first concurrency group is in a running state; and

4           setting a second CG flag in the second concurrency-group data structure to a  
5           value indicating that the second concurrency group is in a running state.

1       6.     The method according to claim 4, further comprising:

2           appending UP-coded workloads enqueued on a first current queue in the first  
3           concurrency-group data structure onto a first active queue in the first concurrency-group  
4           data structure; and

5           appending UP-coded workloads enqueued on a second current queue in the sec-  
6           ond concurrency-group data structure onto a second active queue in the second  
7           concurrency-group data structure.

1       7.     The method according to claim 6, further comprising:

2           dequeueing UP-coded workloads in the first and second concurrency groups from  
3           the first and second active queues, respectively; and

4           executing the dequeued UP-coded workloads to completion.

1       8.     The method according to claim 5, further comprising:

2           in response to the first execution vehicle finishing execution of the UP-coded  
3           workloads in the first concurrency group, the first execution vehicle performing the steps:

4                A)   if at least one UP-coded workload in the first concurrency group is  
5                executable:

6                       (i) setting the value of the first CG flag to a value indicat-  
7                       ing that the first concurrency group is in a queued state;

8                       (ii) re-enqueueing the first concurrency-group data struc-  
9                       ture onto the concurrency-group run queue;

10               B)   if there are not any UP-coded workloads in the first concurrency  
11               group that are executable, setting the first CG flag to a value indicating that the  
12               first concurrency group is in a suspended state;

13                   C) dequeuing from the concurrency-group run queue a third  
14 concurrency-group data structure associated with a third concurrency group; and  
15                   D) setting a third CG flag in the third concurrency-group data structure to  
16 a value indicating that the third concurrency group is in a running state.

1    9.     The method according to claim 1, wherein at least one of the UP-coded workloads  
2 is organized into the one or more concurrency groups at run-time.

1    10.    The method according to claim 1, wherein the MP computer system is a network  
2 cache.

1    11.    A multiprocessor (MP) computer system configured to execute uniprocessor (UP)  
2 coded threads without having to rewrite the UP-coded threads' code, the MP computer  
3 system comprising:

4           a plurality of processors;

5           a memory having a plurality of storage locations addressable by the plurality of  
6 processors for storing data and program code, the memory being configured to store a  
7 separate concurrency-group data structure for each of a plurality of concurrency groups,  
8 each concurrency-group data structure comprising:

9                   an active-queue pointer storing a location in the memory of an active  
10 queue of UP-coded thread messages associated with UP-coded threads in an ex-  
11 ecutable state; and

12                   a current-queue pointer storing a location in the memory of a current  
13 queue of UP-coded thread messages associated with UP-coded threads waiting to  
14 be transferred to the active queue.

1    12.    The MP computer system according to claim 11, wherein each concurrency-group  
2 data structure further comprises a CG flag that stores a value indicating an operational  
3 state of a concurrency group associated with the concurrency-group data structure.

1 13. The MP computer system according to claim 11, wherein each UP-coded thread  
2 message stored in the active queue and current queue stores a location in the memory of a  
3 top of a call stack associated with a specific UP-coded thread.

1 14. The MP computer system according to claim 13, wherein the call stack is accessi-  
2 ble through a thread control block (TCB) associated with the specific UP-coded thread,  
3 the TCB including a CG pointer for storing a memory location of a concurrency-group  
4 data structure.

1 15. The MP computer system according to claim 11, wherein each concurrency-group  
2 data structure further comprises meta-data information associated with a concurrency  
3 group.

1 16. The MP computer system according to claim 11, wherein the MP computer sys-  
2 tem is a network cache.

1 17. An apparatus for executing uniprocessor (UP) coded workloads in a multiproces-  
2 sor (MP) computer system without having to rewrite the UP-coded workloads' code, the  
3 method comprising the steps:

4 means for organizing the UP-coded workloads into one or more concurrency  
5 groups, wherein UP-coded workloads in the same concurrency group are not permitted to  
6 execute concurrently with one another in the MP computer system;

7 means for scheduling first and second execution vehicles that respectively execute  
8 on different processors in the MP computer system at substantially the same time;

9 means for acquiring a first concurrency group by the first execution vehicle;

10 means for acquiring a second concurrency group by the second execution vehicle;

11 and

12 means for executing UP-coded workloads in the first concurrency group through  
13 the first execution vehicle at substantially the same time as UP-coded workloads in the  
14 second concurrency group are executed through the second execution vehicle.

- 1 18. The apparatus according to claim 17, wherein the UP-coded workloads are UP-  
2 coded threads, and the first and second execution vehicles are first and second processes.
- 1 19. The apparatus according to claim 17, wherein the UP-coded workloads are mes-  
2 sages, and the first and second execution vehicles are first and second threads.
- 1 20. The apparatus according to claim 17, further comprising:  
2 means for dequeuing from a concurrency-group run queue a first concurrency-  
3 group data structure associated with the first concurrency group; and  
4 means for dequeuing from the concurrency-group run queue a second  
5 concurrency-group data structure associated with the second concurrency group.
- 1 21. The apparatus according to claim 20, further comprising:  
2 means for setting a first CG flag in the first concurrency-group data structure to a  
3 value indicating that the first concurrency group is in a running state; and  
4 means for setting a second CG flag in the second concurrency-group data struc-  
5 ture to a value indicating that the second concurrency group is in a running state.
- 1 22. The apparatus according to claim 20, further comprising:  
2 means for appending UP-coded workloads enqueued on a first current queue in  
3 the first concurrency-group data structure onto a first active queue in the first  
4 concurrency-group data structure; and  
5 means for appending UP-coded workloads enqueued on a second current queue in  
6 the second concurrency-group data structure onto a second active queue in the second  
7 concurrency-group data structure.
- 1 23. The apparatus according to claim 22, further comprising:  
2 means for dequeuing UP-coded workloads in the first and second concurrency  
3 groups from the first and second active queues, respectively; and  
4 means for executing the dequeued UP-coded workloads to completion.

- 1    24.    The apparatus according to claim 21, further comprising:  
2            means for setting the value of the first CG flag to a value indicating that the first  
3    concurrency group is in a queued state or in a suspended state; and  
4            means for re-enqueueing the first concurrency-group data structure onto the  
5    concurrency-group run queue.
- 1    25.    A computer-readable media comprising instructions for execution in one or  
2    more processors for executing uniprocessor (UP) coded workloads in a multiprocessor  
3    (MP) computer system without having to rewrite the UP-coded workloads' code, the  
4    method comprising the steps:  
5            organizing the UP-coded workloads into one or more concurrency groups,  
6    wherein UP-coded workloads in the same concurrency group are not permitted to execute  
7    concurrently with one another in the MP computer system;  
8            scheduling first and second execution vehicles that respectively execute on differ-  
9    ent processors in the MP computer system at substantially the same time;  
10           acquiring a first concurrency group by the first execution vehicle and a second  
11    concurrency group by the second execution vehicle; and  
12           executing UP-coded workloads in the first concurrency group through the first  
13    execution vehicle at substantially the same time as UP-coded workloads in the second  
14    concurrency group are executed through the second execution vehicle.
- 1    26.    The computer-readable media according to claim 25, wherein the UP-coded  
2    workloads are UP-coded threads, and the first and second execution vehicles are first and  
3    second processes.
- 1    27.    The computer-readable media according to claim 25, wherein the UP-coded  
2    workloads are messages, and the first and second execution vehicles are first and second  
3    threads.
- 1    28.    A method for executing workloads in a multiprocessor (MP) computer system, the  
2    method comprising the steps:

3           organizing the workloads into one or more concurrency groups, wherein work-  
4   loads in the same concurrency group are not permitted to execute concurrently with one  
5   another in the MP computer system;  
6           scheduling first and second execution vehicles that respectively execute on differ-  
7   ent processors in the MP computer system at substantially the same time;  
8           acquiring a first concurrency group by the first execution vehicle and a second  
9   concurrency group by the second execution vehicle; and  
10          executing workloads in the first concurrency group through the first execution ve-  
11   hicle at substantially the same time as workloads in the second concurrency group are  
12   executed through the second execution vehicle.

1   29.    The method according to claim 28, wherein the step of acquiring the first and sec-  
2   ond concurrency groups further comprises:

3           dequeueing from a concurrency-group run queue a first concurrency-group data  
4   structure associated with the first concurrency group; and  
5           dequeueing from the concurrency-group run queue a second concurrency-group  
6   data structure associated with the second concurrency group.

1   30.    The method according to claim 29, further comprising:

2           setting a first CG flag in the first concurrency-group data structure to a value indi-  
3   cating that the first concurrency group is in a running state; and  
4           setting a second CG flag in the second concurrency-group data structure to a  
5   value indicating that the second concurrency group is in a running state.

1   31.    The method according to claim 29, further comprising:

2           appending workloads enqueued on a first current queue in the first concurrency-  
3   group data structure onto a first active queue in the first concurrency-group data struc-  
4   ture; and  
5           appending workloads enqueued on a second current queue in the second  
6   concurrency-group data structure onto a second active queue in the second concurrency-  
7   group data structure.

- 1 32. The method according to claim 31, further comprising:  
2 dequeueing workloads in the first and second concurrency groups from the first  
3 and second active queues, respectively; and  
4 executing the dequeued workloads to completion.
- 1 33. The method according to claim 30, further comprising:  
2 in response to the first execution vehicle finishing execution of the workloads in  
3 the first concurrency group, the first execution vehicle performing the steps:  
4 A) if at least one workload in the first concurrency group is executable:  
5 (i) setting the value of the first CG flag to a value indicat-  
6 ing that the first concurrency group is in a queued state;  
7 (ii) re-enqueueing the first concurrency-group data struc-  
8 ture onto the concurrency-group run queue;  
9 B) if there are not any workloads in the first concurrency group that are  
10 executable, setting the first CG flag to a value indicating that the first concurrency  
11 group is in a suspended state;  
12 C) dequeueing from the concurrency-group run queue a third  
13 concurrency-group data structure associated with a third concurrency group; and  
14 D) setting a third CG flag in the third concurrency-group data structure to  
15 a value indicating that the third concurrency group is in a running state.